Improving Small LLMs performance with RAG

1st Thiago Souza
Centro de Informática
UFPE
Recife – PE, Brazil
tas3@cin.ufpe.br

2nd Cleber Zanchetin
Centro de Informática
UFPE
Recife – PE, Brazil
tas3@cin.ufpe.br

Abstract — Large Language Models (LLMs) have revolutionized access to knowledge, enabling anyone with an internet connection to explore a vast array of topics. However, LLMs are inherently limited by their training data and struggle with subjects beyond their pre-trained knowledge. This challenge is even more pronounced in smaller LLMs, which, due to their reduced parameter count, have a lower capacity to capture complex or niche information. With the growing interest in deploying LLMs on edge devices or local systems, improving their performance is crucial. In this study, we explore how Retrieval-Augmented Generation (RAG) can enhance the inference capabilities of small LLMs by dynamically incorporating relevant context from a vector database. Our research evaluates the extent to which RAG can bridge the performance gap between small and large LLMs, making them more viable for real-world applications.

Keywords — LLM, RAG, Retrieval-Augmented Generation, Small Language Models, Vector Databases

I. BACKGROUND

The introduction of Multi-Head Attention transformer-based Large Language Models (LLMs) [1] revolutionized how people learn, work, and interact with technology. OpenAI popularized these models through their online web chat tool, ChatGPT, making advanced AI-driven conversations accessible to the general public.

Before LLMs, computer-based search primarily relied on string matching or heuristic-based algorithms in platforms like Google and Bing. These approaches, while effective, often required users to formulate precise queries and manually filter through results. The advent of LLMs fundamentally changed this dynamic by enabling users to engage in natural language interactions. Instead of merely retrieving keyword-based results, LLMs interpret queries contextually and generate human-like responses, giving users the impression of conversing with an intelligent assistant.

This transformation not only enhanced search and information retrieval but also expanded how people process and synthesize knowledge. By leveraging attention mechanisms and text embeddings, LLMs facilitate semantic search, allowing users to explore topics more intuitively and extract deeper insights from their queries.

However, running inference-based pipelines on those LLMs is only possible in very powerful data centers as it's massively dependent on a very high number of GPUs and energy availability – which are significant issues in resource constrained environments. LLMs are also very high energy consumers which makes it also very expensive to run on scale. That has created the need of having to explore and design Small Language Models (SM) that, in nature, are similar to LLMs but rely on a much smaller number of parameters hance have more lightweight inference in terms of GPU requirements and energy consumption.

Dimension	LLMs	SMs
Accuracy	state-of-the-art	decent
Generality	general purpose	task specific
Efficiency	resource intensive	resource efficient
Interpretability	low interpretable	high interpretable

Table 1: comparisons of dimensions between LLMs and SMs

Since knowledge is encoded in parameters in language models, it's not surprising that SMs have worse overall performance in inference tasks against their large counterparts. It's also less capable of generalizing and handle broad spectrum of information [2].

There are many techniques such as fine-tuning, knowledge distillation or Retrieval Augmented Generation (RAG) can be used to make SMs more competitive against LLMs. This document aims to explore the use of RAG in the scope of Question / Answer task in fixed knowledge domain as a tool to make SMs a viable option in resource constrained setups.

II. EXPERIMENT AND CONTRIBUTIONS

Our objective is to investigate whether and to which extent RAG can help Small Language Models improve or match performance of larger models.

A. *The experiment*

Our exploration involved the following high-level steps: first we ask a given set of questions to various models, then we evaluate their answers by sending the answer and ground truth to a judge, then we ask the same dataset of questions along with some context data – fetched from a vector database – and process the answers to the judge. The judge itself is a Large Language Model; we selected Open AI's GPT-4o-mini but in future work it's worth exploring more capable models such as Open AI's *o3* models that have higher reasoning capabilities. In this research it wasn't possible due to its high cost per 1M tokens.

The dataset we use in this experiment is the Harry Potter book series and the Harry Potter Trivia Questions [3]

B. *The compared models*

In our research, selected models from various vendors and of various sizes and divided them in four high level buckets: tiny

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

(2B or less parameters), small (3B – 7B parameters), medium (8B – 14B parameters) and large (100B parameters or more).

- Tiny: qwen2.5-0.5b, deepseek-r1-1.5b, qwen2.5-1.5b, gemma3-1b, gemma2-2b.
- Small: llama3.2-3b, llama2-7b.
- Medium: gemma3-12b, deepseek-r1-14b
- Large: GPT-4o-mini

C. The Objective and Contributions

With this experiment we aim to investigate whether RAG can help small LLMs match the performance of larger models in closed domain Question-Answer tasks.

Our contributions are:

- Compare small and large LLMs in a closed-domain QA task (Harry Potter books).
- Evaluate the impact of RAG-enhanced retrieval on smaller models.
- Provide quantitative analysis using multiple evaluation metrics

III. RELATED WORK

The rise of Large Language Models has prompted an active research community to examine not only their capabilities but also the practical trade-offs between model size, domain specialization, and augmentation strategies such as retrieval augmented generation (RAG). This section highlights relevant research exploring the performance of domain-specific LLMs, the role of small models in the LLM ecosystem, and recent findings on the limitations of smaller LLMs in multi-step tasks.

There has been investigations where scientists question the necessity of training specialized domain-specific models in light of the capabilities of general-purpose LLMs like GPT-3.5 and GPT-4 [4]. Some of the findings suggest that general-purpose LLMs, when given sufficient context through prompt engineering or retrieval, can perform competitively with domain-specific models. This supports the idea that retrieval-augmented generation (RAG) may compensate for gaps in internal model knowledge, especially for more generalizable models—an idea central to our study, where we test whether smaller LLMs can similarly benefit from retrieval in a closed-domain QA task. Another study in this space highlights the growing relevance of small LLMs for edge and privacy-sensitive applications [5]. The survey emphasizes techniques such as distillation and RAG to improve their utility without increasing model size — aligning with our focus on enhancing small models through retrieval. There's also studies on how selecting right chunk size improves results of RAG based applications by ensuring optimization in the use of the LLM context window [6].

In contrast, there's also research demonstrating that small models struggle with complex tasks requiring reasoning and tool use. While our setting is simpler — closed-domain QA — this reinforces the motivation to explore whether small LLMs, aided by retrieval, can still perform competitively in targeted applications [7].

IV. METHODOLOGY

A. Dataset and knowledge source

In our research to serve as the knowledge base data we used the Harry Potter Books dataset [8] that contains all seven books of the famous series. The dataset have the books in plain textual form which was a convenient aspect that facilitated the processing step in the knowledge base chunking and ingesting step in the pipeline.

B. Question/Answer dataset

For the question and answer step we relied on the Harry Potter Trivia Questions dataset [9]. The dataset has two files train and test both using the *parquet* file format where each line in the files present a question and an answer. We converted the *parquet* files to a textual form so we could process them as a comma separated value file.

C. Baseline evaluation

In order to serve as a baseline we ran an inference job of a total of 50 questions through the models referred and out of the answers we calculated an average score visible in the Table 2. It's noticeable that most SMs perform quite poorly with the exception of the DeepSeekR1-14B that performs even better than the LLM GPT-4o-mini, which indicates that SMs can indeed be competitive in some closed domain tasks even without the support of techniques like RAG.

Model	**Score**
qwen2.5-0.5b	0.1184
deepseek-r1-1.5b	0.2163
llama2-7b	0.2408
qwen2.5-1.5b	0.2612
gemma3-1b	0.2735
llama3.2-3b	0.3082
gemma2-2b	0.3531
gemma3-12b	0.4939
deepseek-r1-14b	**0.6306**
Small LLMs Average	0.3218
gpt-4o-mini (LLM)	0.5857

Table 2: results of SMs and LLMs

D. The Judge

The score evaluation was given by an LLM (GPT 4o mini) that acted as the judge of how close to the ground truth each answer was in a range that went from 1 to 10 which then got converted to a range between 0 and 1 where 0 means poor answer and 1 means perfect answer. The prompt used in the evaluation process, passed to the judge LLM, provided instructions to the LLM to be impartial and consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and specially how direct the response was [Judge Prompt]. We then calculated the average score for each model assessed.

E. The Question Answerer AI assistant (QAAI)

The AI assistant is the component in the Question and Answer pipeline that receives the question, combines it with an instruction prompt and sends it out for LLM inference. Then the response is received and processed to extract the LLM answer from it. This is the component in the pipeline that works based off of abstractions that allow us to pass a model name and: loads model with such name into the local environment, or, in the case of a Large LLM (such as GPT 4o mini), sends out the query to a cloud based API.

F. Question and Answer (Q&A) pipeline

The Q&A pipeline consists of the steps of: 1/ loading the questions and ground truths dataset and, for each question and ground truth, 2/ sending the question to the QAAI and storing the answer, then 3/ sending the tuple (question, ground truth, answer) to the evaluators, 4/ storing the result aggregated by model name locally for future analysis.

G. Evaluators

For the evaluation step, we have experimented with multiple evaluators such as string distance based of the Levenshtein or Jaro-Winkler and LLM based evaluators.

Model	Jaro Winkler	Levensthein	LLM
deepseek-r1-1.5b	0.4905	0.9906	0.2163
deepseek-r1-14b	0.4818	0.9899	0.6184
gemma2-2b	0.4455	0.7312	0.3531
gemma3-12b	0.3228	0.5828	0.4878
gemma3-1b	0.5547	0.874	0.1021
gpt-4o-mini	0.3304	0.5792	0.5959
llama2-7b	0.5164	0.9299	0.2347
llama3.2-3b	0.5098	0.8344	0.3122
qwen2.5-0.5b	0.586	0.9	0.1184
qwen2.5-1.5b	0.4875	0.8199	0.2633

Table 3 Evaluators results: String distance based versus LLM based score

In the Table 3 it is visible how the LLM based score is misaligned with the String based stores. Doing an in-depth human analysis on how those score evaluators aligned with the human assessment of whether the answer deserved a bad score (close to zero) versus a good score (close to one) the outcome was that the LLM based scoring was much more aligned. Our interpretation is that the nature of the evaluation doesn't allow for string distance based evaluation as with that one loses the semantic aspect of the comparison; as such we decided to rely solely on the LLM based Evaluator described in the section The Judge.

H. RAG pipeline: chunking the data

Once collected the baseline data we then designed the RAG pipeline where the first step we took was to build the process that would chunk the knowledge base data in chunks that could then be ingested by the Vector database of our choice. We decided to have three chunking strategies:

- Chunk size of 600 with overlap of 100 chunks
- Chunk size of 1000 with overlap of 100 chunks
- Chunk size of 1500 with overlap of 100 chunks

The reason was that we wanted to have the ability to experiment with various chunking configurations and measure which one performed better.

I. RAG pipeline: ingesting the knowledge base data

The database of choice was Pinecone [10] which is a very popular and high performing vector database RAG pipeline based applications use but we have also experimented using ChromaDB which is an open source alternative that provided reasonable results [11]. We created three indexes in Pinecone where each one hosted one of the three chunking configurations listed above and shifted between them in our experiments.

The ingestion process involved loading the books corpus in memory and chunking them according to the configuration passed and creating documents in the vector database with a source reference pointing to which book had the context.

J. RAG pipeline: context retrieval

Context retrieval is a key step in a RAG pipeline and the results of our RAG pipeline are highly influenced by the various choices one can make in this step. In our research we made several experiments with different search algorithms, various similarity thresholds and relevant hyper parameters like k that represents the max number of documents the pipeline will retrieve to combine with the prompt [6]. The search algorithms we experimented with were: *similarity_score_threshold* that computes the similarity between the context and the query and assigns a score between 0 and 1 and returns documents that have score above a certain threshold, and *mmr* (Max Marginal Relevance) that combines similarity and diversity.

V. EXPERIMENTS AND RESULTS

A. Baseline performance

As seen in Table 2 most small LLMs performed poorly with the majority of them scoring below 0.3, of a range between 0 and 1, with the exception of the DeepSeek R1 14B model, which was actually the baseline winner, and performed even better than the only Large model we used in this experiment, GPT 4o mini. The Small LLMs average score was 0.3218 showing that, without providing more context for the LLM to perform its inference, the accuracy of the answers was in general not very useful.

B. Impact of RAG

To test our hypothesis, that RAG could make small models be competitive against their large counterparts in domains like the closed domains in question and answer tasks, we executed the RAG pipeline and collected the results which can be seen in the Table 4. With the introduction of RAG in the inference process the average score across small LLMs increased to 0.4272 which represented an increase of 42% compared to average of the small LLMs without the support of RAG as visible in the Table 4. We also see that not all models benefited from having RAG support such as Gemma3-12B that consistently performed worse when RAG support was provided in our experiments. For our surprise, GPT 4o mini also scored worse when RAG support was introduced with a score 21% below the execution with no RAG support.

Model	No RAG	RAG
gemma3-1b	0.1021	0.3000
qwen2.5-0.5b	0.1184	0.3551
deepseek-r1-1.5b	0.2163	0.3571
qwen2.5-1.5b	0.2633	0.3939
gemma3-12b	0.4878	0.4204
llama2-7b	0.2347	0.4245
gemma2-2b	0.3531	0.4327
llama3.2-3b	0.3122	0.4898
deepseek-r1-14b	**0.6184**	**0.6714**
Small LLMs Average	0.3007	0.4272
gpt-4o-mini (LLM)	0.5959	0.4694

Table 4 Small LLMs vs Large LLM with/out RAG

C. Analysis on SMs where performed worse with RAG

We ran an analysis to understand why the model Gemma3-12B with RAG performed worse than without it. Here's some results where the RAG version got a lower score:

- Q1: What sensation accompanies the Sea Urchin Jinx?
- Ground Truth (GT): Discomfort
- No RAG Answer (NRA): A prickly sensation. Score: 0.7
- RAG Answer (RA): I don't know. Score: 0.1
- Q2: Vipera Evanesca is used to vaporize what kind of creature?
- GT: Snake
- NRA: Snakes. Score: 0.7
- RA: I don't know. Score: 0.1
- Q3: Where is Hogwarts located?
- GT: Scotland
- NRA: Scotland. Score: 0.7
- RA: Hogwarts is located in the West Country. Score: 0.2

Based on the results above it appears that in some cases the SM knows the answer but when provided with more context it gets confused and uncertain about whether it knows the answer and follows the instruction that says that *"when you don't know the answer reply with 'I don't know'"* demonstrating that adding context to the query is not always positive to the question and answer process.

Performing the same analysis on Large models we found different characteristics in the mistakes or low score answers:

- Q4: When was Hogwarts founded?
- GT: 10th century
- NRA: Hogwarts was founded around 990 AD. Score: 0.8
- RA: Hogwarts was founded over a thousand years ago. Score: 0.4
- Q5: Harry first took the Knight Bus in The Prisoner of Azkaban. How much does a ticket cost if it includes hot chocolate?
- GT: 14 sickles.
- NRA: A ticket costs 14 Sickles if it includes hot chocolate. Score 1.0
- RA: A ticket for the Knight Bus that includes hot chocolate costs 11 Sickles. Score: 0.3

The results show that providing context in this case actively confused the LLM who would have answered perfectly or with a very high score but was led to provide a less precise answer due to the extra context that was provided by the RAG pipeline.

D. Analysis on SMs where performed better with RAG

In order to analyze how RAG improved the Question-and-Answer process in the scope of small models we analyzed the results of the SM which had the biggest increase of all SMs when RAG was added: Qwen2.5 0.5B which had an increase of 200% going from 0.1184 without RAG to 0.3551 with RAG support. Here's some results:

- Q6: What are the three European wizarding schools that participate in the Triwizard Tournament?
- GT: Hogwarts, Beauxbatons, and Durmstrang.
- NRA: I don't know. Score: 0.1
- RA: The three European wizarding schools that participate in the Triwizard Tournament are: 1. Hogwarts School of Witchcraft and Wizardry 2. Beauxbatons Academy 3. Durmstrang Institute. Score: 1.0
- Q7: What magical creature runs guards wand wood trees?
- GT: Bowtruckles
- NRA: I don't know. Score: 0.1
- RA: The magical creature that runs guards wand wood trees is a bowtruckle. Score: 0.7

The results show that without additional context the model is uncertain about the answer and doesn't provide a valid answer but given additional context, through the RAG support, it answers the question with a very high score which demonstrates that, indeed, RAG can be very useful in setups such the one described in this document.

E. Limitations

One of the limitations in this research was the money resourcing constraints that prevented us from running more powerful models on the evaluation step that could otherwise have been done with a model that supports reasoning such as GPT o3-mini or GPT o1-pro. The scoring provided by The Judge LLM (GPT 4o-mini) were "unfair" in some cases as seen below:

- Q8: Name all seven Weasley children.
- GT: Bill, Charlie, Percy, Fred, George, Ron, and Ginny.
- NRA: Bill, Charlie, Percy, Fred, George, Ron, and Ginny. Score: 1.0
- RA: Bill, Charlie, Percy, Fred, George, Ginny, and Ron. Score: 0.8

The RAG supported answer got a 0.2-lower score compared to the NRA because of the way it ordered the answer in which case the order of the names wasn’t relevant and both answers should get the same high score. In fact, we ran this question and answer by another LLM judge based on the o3-mini (version 2025-01-31) and got the score 1.0 for the NRA answer above. See Instance evaluation through o3-mini in the Appendix for details.

F. Future work

APPENDIX

A. Judge Prompt

```
[Instruction]\nPlease act as an impartial judge
    and evaluate the quality of the response provided by an AI
    assistant to the user question displayed below in a trivia game.
    The question is about something in the world the Harry Potter book series.
    Your evaluation should consider factors such as the helpfulness, relevance,
    accuracy, depth, creativity, and specially how direct the response was.
[Ground truth]\n{reference}\nBegin your evaluation
    by providing a short explanation. Be as objective as possible.
    After providing your explanation, you must rate the response on a scale of 1 to 10
    by strictly following this format: [[rating]], for example: Rating: [[5]].
[Question]\n{input}\n\n[The Start of Assistant\'s Answer]\n{prediction}\n\
[The End of Assistant\'s Answer]',
```

B. Instance evaluation through o3-mini

```
$ python3 llm.py --model gemma3-12b --vector_store_config_name s600-o100 --k 12 --score_threshold 0.6 --
verbose True

Namespace(model='gemma3-12b',vector_store_config_name='s600-o100',k=12,score_threshold=0.6, verbose=True)
> Name all seven Weasley children.
Bill, Charlie, Percy, Fred, George, Ginny, and Ron.
Evaluate? (y/n)
> y
Type the Ground Truth
> Bill, Charlie, Percy, Fred, George, Ron, and Ginny.
[{'Ground_Truth': 'Bill, Charlie, Percy, Fred, George, Ron, and Ginny.', 'Answer': 'Bill, Charlie, Percy,
Fred, George, Ginny, and Ron.', 'Question': 'Name all seven Weasley children.', 'StringDst(JARO)':
0.039215686274509776, 'StringDst(JARO_WINKLER)': 0.02352941176470591, 'StringDst(INDEL)':
0.11764705882352941, 'StringDst(LEVENSHTEIN)': 0.1568627450980392, 'EmbeddingDst(nomic-embed-text)':
0.007110826867603559, 'LabeledScoreString(o3-mini-2025-01-31)': 1.0}]
```